Remarks/Arguments

With reference to the Office Action of July 27, 2007, Applicants offer the following remarks.

Advisory Action of Aril 30, 2007

In the Advisory Action, it was stated that the Reply filed April 12, 2007 failed to place the application in condition for allowance. It was stated that the amendment filed on April 12, 2007 was not entered because the amendments raised new issues that would require further consideration and/or search, and would not place the application in better form for appeal. It was stated that parts of the independent claims are added material and would require further search and consideration. Applicants filed a Request for Continuing Examination to have these new issues considered.

Office Action of July 27, 2007

The specification was objected to as failing to provide a proper antecedent basis for the claimed subject matter, and claims 1,2,10, 14, 18, and 20 were objected to in that there was no concise meaning for the term "axis."

Applicants respectfully traverse the objection for the following reasons:

1. Applicants have amended the independent claims to characterize the axis (singular) – axes (plural) as "XPATH Axes"

2. Support for this limitation is in the specification as filed,

[0004] In an XPath expression, an axis may be specified in addition to a condition for a tag name. If no axis such as /html is specified, for example, an element "child" existing immediately under a node (a child of the node) of a tree structure of a node will be

designated as an axis. In specifying an axis in an XPath expression, an axis can be specified by syntax such as /decendant::p. "decendant" indicates a descendant element. When an axis is defined as /descendant::p, all descendant "p's" within the tree structure can be searched instead of an element existing immediately below. The axis "descendant" can be abbreviated as "//p" or the like for simplicity.

3. Within the art of XPATH the concept of XPATH axes is well known and is even described in Altinel et al. Specifically, Altimel describes a search system that utilizes XPATH[1] and refers to the XPATH standard which characterizes XPATH axes, their names, and their functions thusly:

---

[1] **2.2 XPath as a Profile Language**

The profile model used in XFilter is based on XPath [CD99], a language for addressing parts of an XML document that was designed for use by both the XSL Transformations (XSLT) [Cla99b] and XPointer [DDM99] languages. XPath provides a flexible way to specify path expressions. It treats an XML document as a tree of nodes; XPath expressions are patterns that can be matched to nodes in the XML tree. The evaluation of an XPath pattern yields an object whose type can be either a node set (i.e., an unordered collection of nodes without duplicates), a boolean, a number, or a string.

Paths can be specified as absolute paths from the root of the document tree or as relative paths from a known location (i.e., the context node). A query path expression consists of a sequence of one or more *location steps*.In the simplest and most common form, a location step specifies a node name (i.e., an element name). (Footnote in original: *The full XPath specification [CD99] contains many more options. We do not list them all here due to space considerations.*)

The hierarchical relationships between the nodes are specified in the query using parent-child ("/") operators (i.e., at adjacent levels) and ancestor-descendant ("//") operators (i.e., separated by any number of levels). For example the query /catalog/product//msrp addresses all msrp element descendants of all product elements that are direct children of the catalog (root) element in the document. XPath also allows the use of a wildcard operator ("*"), which matches any element name, at a location step in a query.

Each location step can also include one or more *filters* to further refine the selected set of nodes. A filter is a predicate that is applied to the element(s) addressed at that location step. All the filters at a location step must evaluate to

TRUE in order for the evaluation to continue to the descendant location steps. Filter expressions are enclosed by "[" and "]" symbols. The filter predicates can be applied to the text of the addressed elements or the attributes of the addressed elements and may also include other path expressions. Any relative paths in a filter expression are evaluated in the context of the element nodes addressed in the location step at which they appear. For example, consider the query: //product[price/msrp < 300]/name. This query selects the name elements of the XML document if the msrp of the product is less than 300. Here, the path expression price/msrp in the filter is evaluated relative to the product elements. This example also shows how element contents can be examined in the queries.

In XFilter, XPath is used to select entire documents rather than parts of documents. That is, we treat an XPath expression as a predicate applied to documents. If the XPath expression matches at least one element of a document then we say that the document satisfies the expression.

2.2 Axes
The following axes are available:

the child axis contains the children of the context node

the descendant axis contains the descendants of the context node; a descendant is a child or a child of a child and so on; thus the descendant axis never contains attribute or namespace nodes

the parent axis contains the parent of the context node, if there is one

the ancestor axis contains the ancestors of the context node; the ancestors of the context node consist of the parent of context node and the parent's parent and so on; thus, the ancestor axis will always include the root node, unless the context node is the root node

the following-sibling axis contains all the following siblings of the context node; if the context node is an attribute node or namespace node, the following-sibling axis is empty

the preceding-sibling axis contains all the preceding siblings of the context node; if the context node is an attribute node or namespace node, the preceding-sibling axis is empty

the following axis contains all nodes in the same document as the context node that are after the context node in document order, excluding any descendants and excluding attribute nodes and namespace nodes
the preceding axis contains all nodes in the same document as the context node that are before the context node in document order, excluding any ancestors and excluding attribute nodes and namespace nodes

the attribute axis contains the attributes of the context node; the axis will be empty unless the context node is an element

the name space axis contains the namespace nodes of the context node; the axis will be empty unless the context node is an element

the self axis contains just the context node itself
the descendant-or-self

axis contains the context node and the descendants of the context node

the ancestor-or-self axis contains the context node and the ancestors of the context node; thus, the ancestor axis will always include the root node

NOTE: The ancestor, descendant, following, precedingand self axes partition a document (ignoring attribute and namespace nodes): they do not overlap and together they contain all the nodes in the document.

Axes
[6] *AxisName ::= 'ancestor'*
| *'ancestor-or-self'*
| *'attribute'*
| *'child'*
| *'descendant'*
| *'descendant-or-self'*
| *'following'*
| *'following-sibling'*
| *'namespace'*
| *'parent'*
| *'preceding'*
| *'preceding-sibling'*
| *'self'*

Thus, the amendment is proper, is not new matter, is supported by the specification, and by the general state of knowledge in the art.

Art Rejections

35 USC §103, Altinel et al. in view of Fernandez et al.

All of the claims were rejected under 35 USC §103 as unpatentable over Altinel and Franklin, "Efficient Filtering of XML Documents for Selective Dissemination of Information" in view of United States Patent 6,052,686 of Fernandez et al. for Database Processing Using Schemas.

Art of Record

1. Altinel and Franklin, Efficient Filtering of XML Documents for Selective Dissemination of Information, states that information dissemination applications are gaining increasing popularity due to dramatic improvements in communications bandwidth

and ubiquity. They next describe the problem that the sheer volume of data available drives the use of selective approaches to dissemination in order to avoid overwhelming users with unnecessary information. Existing mechanisms for selective dissemination typically rely on simple keyword matching or "bag of words" information retrieval techniques. The advent of XML as a standard for information exchange and the development of query languages for XML data enables the development of more sophisticated filtering mechanisms that take structure information into account. Altinel and Franklin describe several index organizations and search algorithms that they have developed for performing efficient filtering of XML documents for large-scale information dissemination systems. In the cited paper they describe these techniques and examine their performance across a range of document, workload, and scale scenarios.

Specifically cited is the passage at page 55, column 2,

> "In contrast, in SDI systems, large numbers of queries are stored, and the documents are individually matched to the queries. Thus, in an SDI system, it is necessary to *index the queries*. XFilter is unique in that it combines the scalable SDI approach of indexing queries with the ability to reference document structure (i.e., schema information) leading to scalable but precise filtering of documents for Internet-scale systems. "

and

> "The main structures used in the Filter Engine are depicted in Figure 3. Each XPath query is decomposed into a set of *path nodes* by the XPath parser. These path nodes represent the element nodes in the query and serve as the states of the FSM for the query. Path nodes are not generated for wildcard ("*") nodes."

And

> "These profiles are "standing queries", which are (conceptually) applied to all incoming documents."

Where the characterization of Altinels filter engine is that

> "The Query Index is built over the states of the XPath queries."

Further citing

> "The other key inputs to an SDI system are the documents to be filtered."

And

> "When a document arrives at the Filter Engine, it is run through an XML Parser which then drives the process of checking for matching profiles in the Index. We use an XML

> parser that is based on the SAX interface, which is a standard interface for *event-based* XML parsing [Meg98]. We developed the parser using the *expat* toolkit [Cla99a], which is a non-validating XML processor. "

were used to show the unpatentability of claim 1.

It was specifically stated in the Office Action that

> Altinel does not explicitly disclose "wherein the compiling device generates and registers a state transition by replacing an axis including an axis in the opposite direction and a logical expression including a conjunction or a negative expression while keeping an input query equal in terms of search, wherein the compiling device generates a query automaton including a plurality of states of the backward nodes, a condition for transition, and at least a search state,"

And

> Said query automaton evaluator determining a state transition of a node under determination by storing a left node and a lower node in correspondence with an identified element identifier, and evaluating said query automaton with a search result of said left node and said lower node,".

Fernandez (below) was then cited to overcome this deficiency.

2. United States Patent 6,052,686 of Fernandez et al. for Database Processing Using Schemas describes an apparatus and method for processing a database. The structure of the database is constructed into a schema which only includes those structures of the database that are known. Desired information to be extracted from the database is specified using path expressions and automaton models the path expression. A composite automaton is generated based on the automaton and the schema. The composite automaton is pruned and portions of the database corresponding to the pruned automaton are searched to obtain the desired information. Points within the database may be identified to begin searching for the desired information. These points correspond to states of the composite automaton. A hybrid automaton may be formed for each set of possible starting states to determine completeness by simulating the hybrid automaton against the composite automaton.

Discussion

Status of the Claims.

Claims 1-22 were originally presented for Examination. All of the claims were rejected in the Office Action of December 12, 2006. Subsequently, the claims were amended, and claims 1, 3, 4, 6, 10, 11, 14, 15, 18-20, and 22 were presented for examination and rejected.

Exemplary Claim

Claim 1, as amended, is exemplary.

> *1. A document-searching system for searching a document having a hierarchical structure with elements separated by element identifiers, comprising:*
>
> *a compiling device for generating an XPATH query automaton by storing an input query expression, performing parsing, identifying different types of nodes in said element identifiers, wherein the compiling device generates and registers a state transition by replacing an XPATH axis including an XPATH axis in the opposite direction and a logical expression including a conjunction or a negative expression while keeping an input query expression equal in terms of search, wherein the compiling device generates a query automaton including a plurality of states of the backward nodes, a condition for transition, and at least a search state;*
>
> *a query automaton storage device for storing the query automaton generated by said compiling device;*
>
> *a query automaton evaluator for reading out said query automaton from said storage device and storing said automaton, while reading in said document and performing a stream search by using states of a plurality of different types of nodes in said element identifiers included in said document and said query automaton, said query automaton evaluator determining a state transition of a node under determination by storing a left node and a lower node in correspondence with an identified element identifier, and*

evaluating said query automaton with a search result of said left node and said lower node, and outputting the searched node and

a search result storage means for storing the output of the query automaton evaluator, and for thereafter outputting the stored output of the query automaton evaluator and the output of the searched node.

Discussion of patentability will be with respect to claim 1, since the newly added limitations are common to all of independent claims.

Discussion: Art Rejection

The overarching issue is whether the claims, as limited by the newly added clauses and limitations are allowable over the art of record.

All of the claims either contain one of the two above claim limitations or are dependent on base claims which contain one of the above limitations.

Claim 1 will be analyzed in detail. Amended claim 1 recites:

Analyzing the claim in detail, the claim requires:

A document-searching system
a hierarchical structure
elements separated by element identifiers,

a compiling device
for generating an XPATH query automaton
by storing an input query expression,
performing parsing,
identifying different types of nodes in said element identifiers,
wherein the compiling device generates and registers a state transition
by replacing an XPATH axis including an XPATH axis in the opposite direction and
a logical expression including
a conjunction or a negative expression
while keeping an input query expression equal in terms of search,
wherein the compiling device generates a query automaton including
a plurality of states of the backward nodes,
a condition for transition, and
at least a search state;

a query automaton storage device
for storing the query automaton generated by said compiling device;

a query automaton evaluator for
reading out said query automaton from said storage device and
storing said automaton,
while reading in said document and performing a stream search by
using states of a plurality of different types of nodes in said element identifiers included in said document and said query automaton,
said query automaton evaluator determining a state transition of a node under determination by
storing a left node and a lower node
in correspondence with an identified element identifier, and
evaluating said query automaton with a search result of said left node and said lower node, and
outputting the searched node and

a search result storage means
storing the output of the query automaton evaluator, and
outputting
the stored output of the query automaton evaluator and
the output of the searched node.

In order to be an "anticipation" of the claimed invention, a reference must show each and every element of the claim. Altinel et al. fail this standard. The reference does not teach or even suggest the newly added claim limitations of

> *wherein the compiling device generates and registers a state transition by replacing an axis including an axis in the opposite direction and a logical expression including a conjunction or a negative expression while keeping an input query expression equal in terms of search, wherein the compiling device generates a query automaton including a plurality of states of the backward nodes, a condition for transition, and at least a search state*

and

> *said query automaton evaluator determining a state transition of a node under determination by storing a left node and a lower node in correspondence with an identified element identifier, and evaluating said query automaton with a search result of said left node and said lower node,*

Altinel et al fail this test, reciting:

> The key insight to building high-performance, scalable SDI systems is that in such systems, the roles of queries and data are reversed [YM94]. In a database system large numbers of data items are indexed and stored, and queries are individually applied. In contrast, in SDI systems, large numbers of queries are stored, and the documents are individually matched to the queries. Thus, in an SDI system, it is necessary to *index the queries*. XFilter is unique in that it combines the scalable SDI approach of indexing queries with the ability to reference document structure (i.e., schema information) leading to scalable but precise filtering of documents for Internet-scale systems.

and

> Each XPath query is decomposed into a set of *path nodes* by the XPath parser. These path nodes represent the element nodes in the query and serve as the states of the FSM for the query. Path nodes are not generated for wildcard ("*") nodes. A path node contains the following information:

and

> The process of filtering and delivering documents based on user interests is sometimes referred to as Selective Dissemination of Information (SDI). Figure 1 shows a generic architecture for an XML-based SDI system. There are two main sets of inputs to the system: user profiles and data items (i.e., documents). User profiles describe the information preferences of individual users. In most systems these profiles are created by the users, typically by clicking on items in a Graphical User Interface. In some systems, however, these profiles can be learned automatically by the system through the

> application of machine learning techniques to user access traces. The user profiles are converted into a format that can be efficiently stored and evaluated by the Filter Engine. These profiles are "standing queries", which are (conceptually) applied to all incoming documents.

and finally

> In XFilter, XPath is used to select entire documents rather than parts of documents. That is, we treat an XPath expression as a predicate applied to documents. If the XPath expression matches at least one element of a document then we say that the document satisfies the expression.

These passages do not anticipate the claimed invention. In fact, they teach away from Applicants' claimed invention by indexing queries and reversing queries and documents.

Fernandez et al. recites

> This invention provides an apparatus and method for efficiently processing information contained in an intra-network by treating the intra-network as a database and either pruning the database or identifying a point within the database from which to start processing. Known portions of the database structure is modeled as a schema. The schema only includes those structures of the database that are known and accommodates other structures that are unknown.
>
> A desired information to be extracted from the database is specified using a path expressions and an automaton models the path expression. A composite automaton is generated based on the automaton and the schema. The composite automaton is analyzed and pruned by removing broken paths which do not lead to the desired information. The pruned composite automaton corresponds to that portion of the database that must be processed to obtain the desired information. Thus, processing time is reduced because portions of the database not leading to the desired information are not processed.

With a different application of automatons and path models (using an "intranetwork" analogous to an organization chart) as a database. This is totally different from searching a large quantity of XML type documents, as claimed by applicants.

Altinel et al., either alone or taken with Fernandez et al does not teach or suggest Applicants'claimed invention of—

A document-searching system for searching a document having a hierarchical structure with elements separated by element identifiers, comprising:

a compiling device for generating a query automaton by storing an input query expression, performing parsing, identifying different types of nodes in said element identifiers, wherein the compiling device generates and registers a state transition by replacing an axis including an axis in the opposite direction and a logical expression including a conjunction or a negative expression while keeping an input query expression equal in terms of search, wherein the compiling device generates a query automaton including a plurality of states of the backward nodes, a condition for transition, and at least a search state;

a query automaton storage device for storing the query automaton generated by said compiling device;

a query automaton evaluator for reading out said query automaton from said storage device and storing said automaton, while reading in said document and performing a stream search by using states of a plurality of different types of nodes in said element identifiers included in said document and said query automaton, said query automaton evaluator determining a state transition of a node under determination by storing a left node and a lower node in correspondence with an identified element identifier, and evaluating said query automaton with a search result of said left node and said lower node, and outputting the searched node and

a search result storage means for storing the output of the query automaton evaluator, and for thereafter outputting the stored output of the query automaton evaluator and the output of the searched node.

With the further limitations of

wherein the compiling device generates and registers a state transition by replacing an axis including an axis in the opposite direction and a logical expression including a conjunction or a negative expression while keeping an input query expression equal in terms of search, wherein the compiling device generates a query automaton including a plurality of states of the backward nodes, a condition for transition, and at least a search state

and

said query automaton evaluator determining a state transition of a node under determination by storing a left node and a lower node in correspondence with an identified element identifier, and evaluating said query automaton with a search result of said left node and said lower node,

.

Conclusion

Based on the above discussion, it is respectfully submitted that the pending claims describe an invention that is statutory subject matter and is properly allowable to the Applicants.

If any issues remain unresolved despite the present amendment, the Examiner is requested to telephone Applicants' Attorney at the telephone number shown below to arrange for a telephonic interview before issuing another Office Action.

Applicants would like to take this opportunity to thank the Examiner for a thorough and competent examination and for courtesies extended to Applicants' Attorney.

	Respectfully Submitted
Certificate of Electronic Filing I hereby certify that this paper (along with any paper referred to as being attached or enclosed) is being electronically deposited with the United States Patent and Trademark Office on the date shown below. Date of deposit: January 22, 2008 Person mailing paper: Richard M. Goldman Signature: /Richard M. Goldman/	/s/ Richard M. Goldman January 22, 2008 Richard M. Goldman, Reg. # 25,585 371 Elan Village Lane, Suite 209 San Jose, CA 95134 Voice: 408-324-0716 Fax: 408-324-0672 E-mail: goldmanptn@aol.com